3172 | Lab 3 | 1

Coding style

- object-oriented
- procedural patterns
- data abstraction
- Google C++ guidelines
  - Self-Contained Headers
  - Names and order of Includes
  - Scoping - ~~Names~~ Namespaces
  - Local Variables
  - Static and Global Variables
- Format all output
- multiple input files specified from terminal
- Branch and merge to add new features
- Dependencies - kept in submodules

- use structs
- use classes

File Management

- keep the submodule as a sub directory named "data"
  - All data in "data"
- Three CSV files; 1115, 3115, 3130
- source kept in "src" folder
- Main function kept alone in main.cpp
- Functions kept seperately
  - one file to keep functions for file reading and writing
  - one file to keep functions for aggregating
- Declarations and prototypes kept in headers
- Reports kept in "output"
  - include timestamp in filename
  - Each file keeps only a single report

Suggested structs

Student(ID, Course, Grade)

Data analysis

- pass rate per instructor and per course number

Instructors | pass rate

Ins | p_rate Ins | p_rate Ins | p_rate ...

Course Number | pass Rate

crsN | p_rt crsN | p_rt crsN yyyy

- w rate

Ins | w_rt ins | w_rt ... and crsN | w_rate crsN | w_rate crsN | w_rate

- Fall vs Spring pass rate for each crsN

crsN

fall_rate spring_rate ...

3142 | Lab 3 | 2

Data Files: 1115 | 3115 | 3130

Extracting data for analysis

1. Pass rate per instructor

Course File → Extract Grades → Grades$_{crs}\{g_1, g_2, \ldots, g_k\}$ →

→ Count passing grades → Grades{...} numOf_pass

→ calculate pass rate → numOf_pass / (Grade.size)

(For course pass rate)

options for interpreting

A. Instructor
   - crs
   - crs
   - crs

B. crs
   - ...
   - ins
   - ins

C. ins ...
   - rate

This does not need a ~~struct~~ or class, but for the sake of the assignment I will ~~use those~~ employ both.

Follow interpretation A.

~~Course File → collection of student structs~~

All Course Files → ~~collection~~ set of ~~course~~ instructors

ers file → collect the grades for each instructor ~~and~~ & ~~collect them~~ & label the collection with the course number

~~→ calculate and print the pass~~

calculate and print pass rate for each instructor for each course they taught

2

Now to match instructor with grade collection →

Teacher
List of taught classes

class
crsN
section
term

class

Student
class
List of classes taken paired with their grades

- "student" should not be a struct, as student information is not required

- student and term and section uniquely identify a record

course
History of classes
→ pairs of <section, term>

class

Starting over

Teacher
<class, class, ...>

Student
<<class, g>, <class, g>, ...>

class
crsN
term
section
~~instructor~~ instructor

Instructor
id

Student
<<class, g>, ...>

3142 | Lab3 | 3

- College
  - Instructor Roster
  - student Roster
  - ~~Course~~ History
    class

- Teacher
  - ID

- Student
  - ID
  - classes taken
    ↳ <class, grade>, ... >

- class
  crsN
  term
  section
  teacher_id

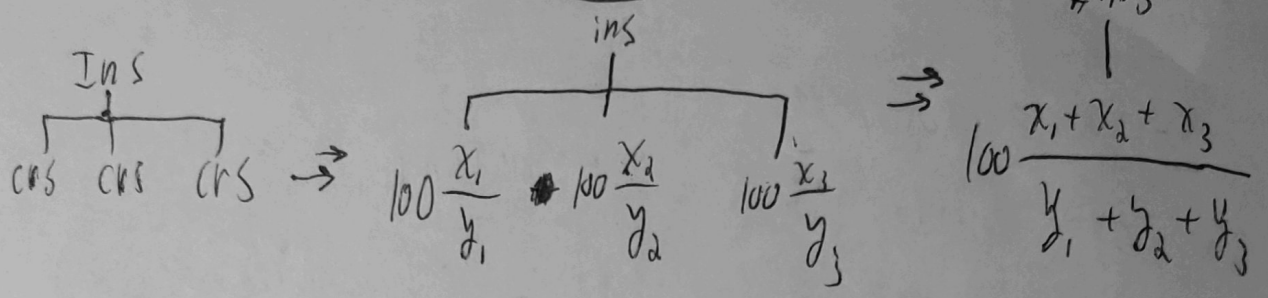
1. Iterating through the instructor roster, match each instructor ID with all instances of it in the class history.

2. ~~For each match,~~ Then match the class with all instances of itself in the student roster (an instances being located in an individual student's class history).

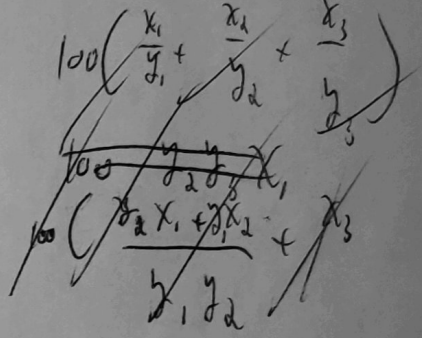
3. Collect each grade into a course-specific collection.

3|

Then to combine the pass rates



Methods Neccessary



- obtain a list of all classes taught by an instructor
- obtain a list of all grades earned in a class
- ~~Collect~~ ~~Collect~~ ~~Organize~~ ~~Organize~~ all grades earned under an instructor into course-specific collections
- collect all grades earned under an instructor into one collection
- calculate the pass rate from a collection of grades

3142 | Lab 3 | 4

~~Extracting~~ Data analysis

Pass rate per course

1. Iterate through the classes and organize them into course-specific collections

~~2. Calculate pass rates/ methods needed~~

2. match each ~~III~~ class with ~~a~~ all instances of itself in ~~the~~ student roster; and collect the grades ~~in~~ ~~for~~ a class into a course-specific collection

crs	crs	crs
|	|	|
pRate	sRate	sRate

I cut off the speeding test ~~since~~ because I ~~a~~ want to ~~revise~~ revise the data structures

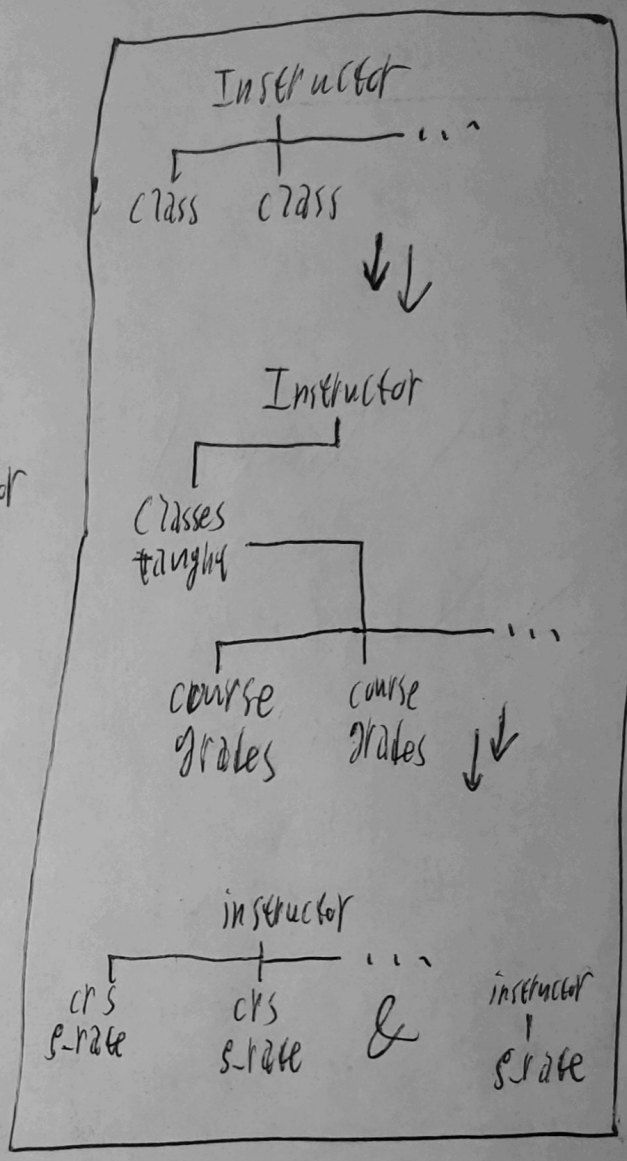
- College
  - Instructor Roster
  - Student Roster
  - Class History
- Instructor
  - ID
- Student
  - ID
- Class
  - crsN
  - term
  - section
  - instructor_ID
  - class_roster → class_roster: <<student_ID, grade>, ...>

Pass rate per instructor 4

1. For each instructor, build a list of classes taught

2. organize the grades in a class roster into a course-specific collection of grades for each instructor

3. Calculate and print pass rates



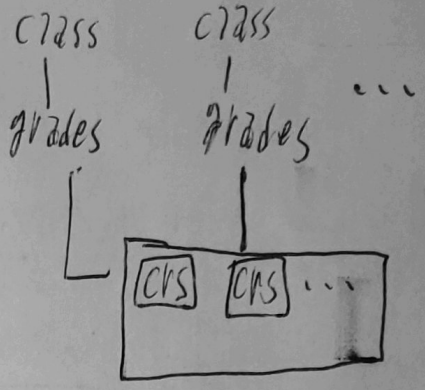
Methods Needed

- Build a list of all classes taught by an instructor
- ~~do~~ ~~collect~~ organize all grades earned under an instructor into course-specific collections
- Collect all grades earned under an instructor into one collection
- Calculate the pass rate from a collection of grades

3142	Lab 3	5

Pass rate per course

1. Iterate through the class history and ~~collect~~ collect the grades of ~~every~~ each class into course-specific collections



2. Calculate and print

Methods Needed

- Collect ~~the~~ grades from every class into ~~a~~ course-specific collections ~~for this for all classes~~

W rate per instructor and per course

Same as pass rate; just use different criteria when ~~to~~ calculating

Fall vs Spring pass rate per course

1. Iterate through the class history and collect the grades into ~~a~~ collections that are both course-specific and term-specific

2. Calculate and print.

Starting over | 5

Pass rate per instructor

Working at the singular level:

1. Build a list of classes taught
2. Iterating through this list, collect the course-specific grades
3. calculate and print the pass rate from this collection
4. Repeat for each existing course
5. collect all grades and calculate overall pass rate
6. Repeat for each instructor

- For this I need a reference of all courses to be checked. This can be derived from a course listing or by building a set of courses from the list of classes

This can be done without intermediary collections; but object-oriented style is requested

Methods Needed

- Build a list of all classes taught by an instructor
- Build a set of courses from a list of classes
- Collect all course-specific grades from a list of classes, given a course
- calculate the pass rate of a collection of grades
- Collect all grades from a list of classes
- Grading Scale

pass rate per course

1. Build a set of courses from the class history
2. ~~For each course,~~ For a given course, collect the course-specific grades from the class history
3. Calculate and print the ~~course~~ pass rate from this collection
4. Repeat for each course

Methods Needed

~~Methods Needed~~

- ~~collect the grades from a list of classes that~~ From a list of classes, collect the grades that correspond to a given course and term

Methods Needed

~~Everything needed has been specified previously~~

w rate

Merely change the criteria in counting the grades

Fall vs Spring pass rate per course

1. Build a set of courses from the class history
2. Collect the grades from the class history that correspond to a given course and term
3. Calculate and print the pass rate from this collection ~~to this for both terms~~
4. ~~Repeat for both terms~~ Do this ~~Repeat for each course~~
4. Do this for both terms
5. Repeat for each course

- College
  - Instructor Roster
  - Student Roster
  - Class History
  - Enrollment History

- Instructor
  - ID
- Student
  - ID
- Class
  - crsN
  - term
  - section
  - instructor_id
- Enrollment
  - class — <studentID, grade>, <studentID, grade>, ...
  - student — <class, grade>, <class, grade>, ...

Source Files

college.cpp
college.h
main.cpp

File structuring

Structs to represent ~~records~~ entities derived from the records : ~~college.cpp~~ college.h

Class that models the environment of the entities : college.h

Implementation of its functions : college.cpp

Processing input and outputting the data analysis : main.cpp

File reading and writing : FileParser.cpp FileParser.h

Aggregating functions for counting : aggregate.cpp aggregate.h

Short definitions

ex / lq

```
class College
{
    Set instructor_Roster
    set student_roster
    Set class_history
    struct enrollment_history

    // Functions to add to rosters and histories

    // Function to check for pass or fail

    // Function to determine the pass rate of a collection
       of grades

    // Function to determine the w rate of a collection of
       grades

    // Function to build a set of courses from a list of
       of classes

    // Function to build a set of classes taught by a
       given instructor

    // Function to collect all grades from a list of
       classes

    // Function to collect all grades for a given course
       from a list of classes

    // Function to collect all grades for given course and
       term from a list of classes

};
```

```
struct Instructor
{
    int id;
    Instructor(int id);
};
```

```
struct Student
{
    int id;
    Student(int id);
};
```

```
struct Class
{
    int crs_num;
    string term;
    string section;
    int instructor_id;
    Class(int crs_num, string term,
          string section, int instructor_id
          );
};
```

```
struct Enroll_History
{
    set <<Class, student>>
    map<Class, vector<Student, grade>> ... ;
    map<Student, Class> ... ;
    map<Student, vector<Class, grade>> ... ;
};
```

← define the method of insertion inside the struct

Reading and Writing

Main opens data file
or
Main passes file name

1. Main shall pass the filename and a "College" object also
2. read function goes through the file line-by-line and populates the "college" object

...I will leave the details of this file for when I code it

8142 | Lab3 | 8

order of ~~writing~~ writing code

1. college.h: ~~first~~ define the structs, ~~and~~ the members of the class, ~~and~~ and declare the file parsing functions
2. college_recordParser.cpp: define the file parsing functions
3. college.cpp: define the class functions ~~for manipulating its members~~
4. main.cpp: data analysis

college_recordParser.cpp

~~Record parsing consists of~~

Record parsing:

- file reading
- line parsing
- validated data entry